FIG. 1

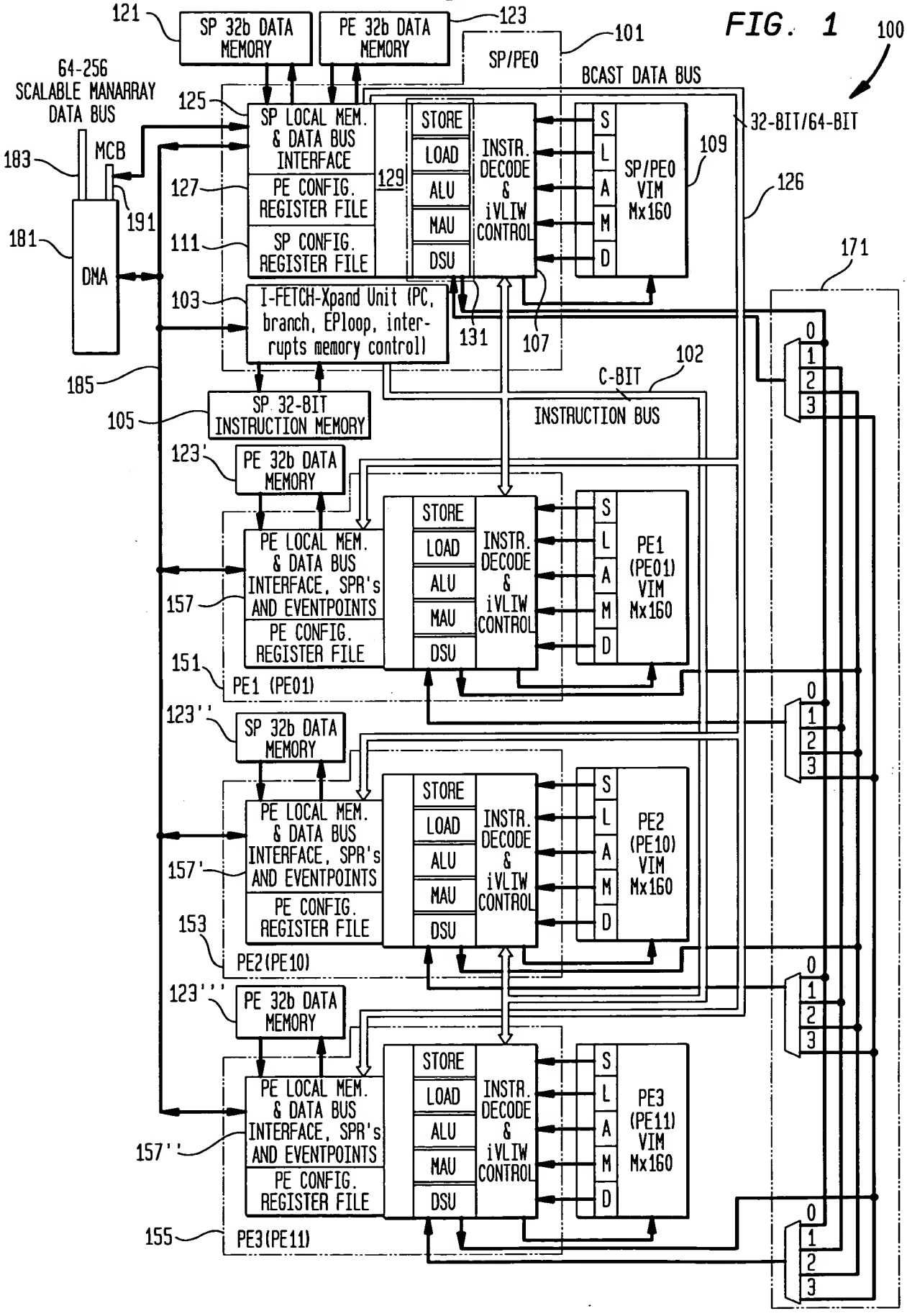

100
121
SP 32b DATA MEMORY
123
PE 32b DATA MEMORY
101
SP/PE0
BCAST DATA BUS
64-256 SCALABLE MANARRAY DATA BUS
125
SP LOCAL MEM. & DATA BUS INTERFACE
127
PE CONFIG. REGISTER FILE
111
SP CONFIG. REGISTER FILE
129
STORE
LOAD
ALU
MAU
DSU
INSTR. DECODE & iVLIW CONTROL
S
L
A
M
D
SP/PE0 VIM Mx160
109
32-BIT/64-BIT
126
171
183
MCB
191
181
DMA
103
I-FETCH-Xpand Unit (PC, branch, EPloop, interrupts memory control)
131
107
102
C-BIT
INSTRUCTION BUS
185
105
SP 32-BIT INSTRUCTION MEMORY
123'
PE 32b DATA MEMORY
PE LOCAL MEM. & DATA BUS INTERFACE, SPR's AND EVENTPOINTS
157
PE CONFIG. REGISTER FILE
PE1 (PE01) VIM Mx160
151
PE1 (PE01)
123''
SP 32b DATA MEMORY
157'
153
PE2 (PE10) VIM Mx160
PE2(PE10)
123'''
PE 32b DATA MEMORY
157''
PE3 (PE11) VIM Mx160
155
PE3(PE11)
0
1
2
3

FIG. 2A

200

LV Encoding

31 30	29	28 27 26 25	24	23 22	21 20 19 18	17	16	15	14	13	12	11	10	9	8	7 6 5 4 3 2 1 0
Group	S/P	CtrlOp	E/D	UAF	InstrCnt	0	0	0	SU	LU	ALU	MAU	DSU	Vb	0	VIMOFFS

FIG. 2B

210

LV Syntax/Operation

```
Instruction Operands          Operation
LV.[SP]     V[01], VIMOFFS,   (V[01]+VIMOFFS)[SU].enable ← 0 if (D = S)
            InstrCnt,         (V[01]+VIMOFFS)[LU].enable ← 0 if (D = L)
            D = {SLAMD},      (V[01]+VIMOFFS)[ALU].enable ← 0 if (D = A)
            F = [AMDN]        (V[01]+VIMOFFS)[MAU].enable ← 0 if (D = M)
                              (V[01]+VIMOFFS)[DSU].enable ← 0 if (D = D)

                              (V[01]+VIMOFFS)[UAF] ← ALU if (F = A or F =)
                              (V[01]+VIMOFFS)[UAF] ← MAU if (F = M)
                              (V[01]+VIMOFFS)[UAF] ← DSU if (F = D)
                              (V[01]+VIMOFFS)[UAF] ← None if (F = N)

                              for (i = 0;i < InstrCnt;i++){
                                 Load instruction into (V[01]+VIMOFFS)
                                 if (SU Instr AND D! = S){(V[01]+VIMOFFS)[SU].enable ← 1}
                                 if (LU Instr AND D! = L){(V[01]+VIMOFFS)[LU].enable ← 1}
                                 if (ALU Instr AND D! = A){(V[01]+VIMOFFS)[ALU].enable ← 1}
                                 if (MAU Instr AND D! = M){(V[01]+VIMOFFS)[MAU].enable ← 1}
                                 if (DSU Instr AND D! = D){(V[01]+VIMOFFS)[DSU].enable ← 1}
                              }
```

FIG. 3A

XV Encoding

300

31 30	29	28 27 26 25	24	23 22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7 6 5 4 3 2 1 0
Group	S/P	CtrlOp	VX	UAF	0	0	0	0	0	0	0	SU	LU	ALU	MAU	DSU	Vb	0	VimOffs

FIG. 3B

XV Syntax/Operation

310

Instruction	Operands	Operation
XV.[SP]	V[01], VIMOFFS,	Execute(V[01]+VIMOFFS)[SU] if (E = S)
	E = {SLAMD}, F = [AMDN]	Execute(V[01]+VIMOFFS)[LU] if (E = L)
		Execute(V[01]+VIMOFFS)[ALU] if (E = A)
		Execute(V[01]+VIMOFFS)[MAU] if (E = M)
		Execute(V[01]+VIMOFFS)[DSU] if (E = D)
		(V[01]+VIMOFFS)[UAF] ← ALU if (F = or F = A)
		(V[01]+VIMOFFS)[UAF] ← MAU if (F = M)
		(V[01]+VIMOFFS)[UAF] ← DSU if (F = D)
		(V[01]+VIMOFFS)[UAF] ← None if (F = N)

FIG. 4A

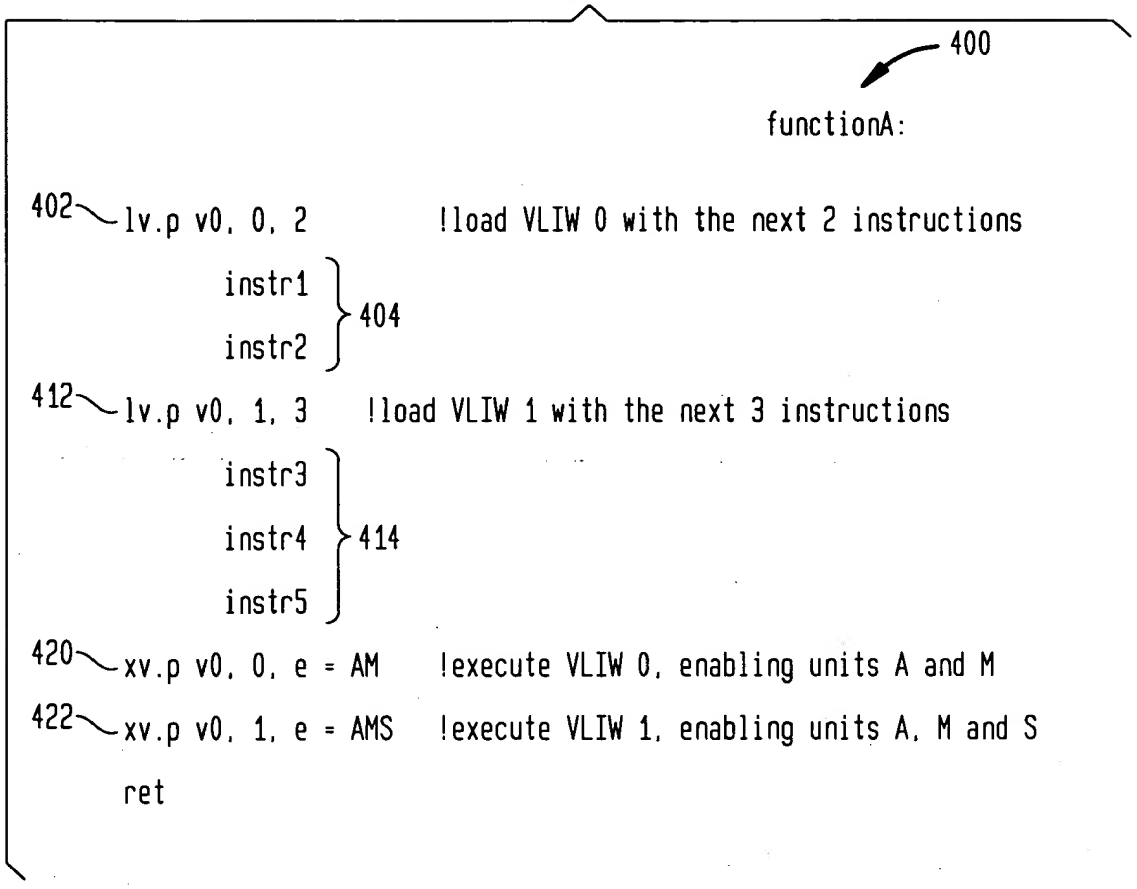

400
functionA:
402
lv.p v0, 0, 2 !load VLIW 0 with the next 2 instructions
instr1
instr2
404
412
lv.p v0, 1, 3 !load VLIW 1 with the next 3 instructions
instr3
instr4
instr5
414
420
xv.p v0, 0, e = AM !execute VLIW 0, enabling units A and M
422
xv.p v0, 1, e = AMS !execute VLIW 1, enabling units A, M and S
ret

FIG. 4B

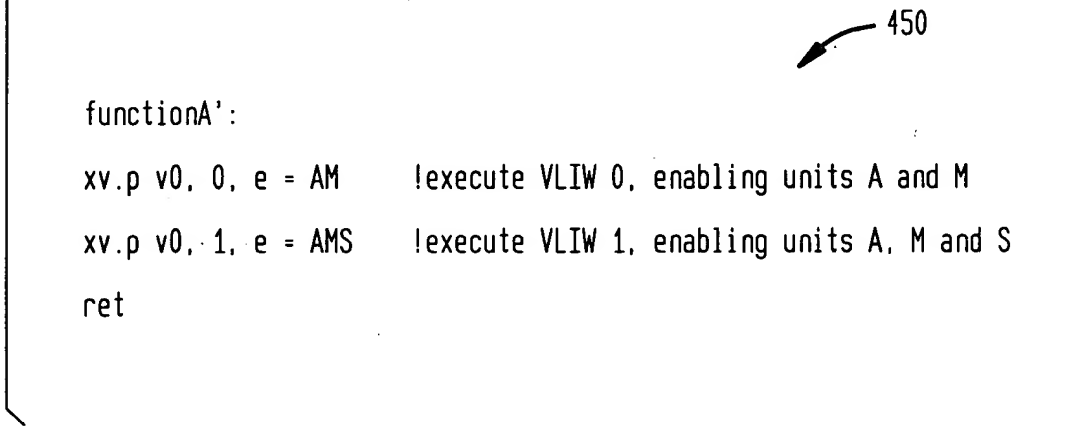

450
functionA':
xv.p v0, 0, e = AM !execute VLIW 0, enabling units A and M
xv.p v0, 1, e = AMS !execute VLIW 1, enabling units A, M and S
ret

FIG. 5

```
                                    500

510-0: Program start

511-1:loop 10 times
512-     executive VLIW a

513-     if condition then

514-2:      executive VLIW b

515-3:   else
516-        execute VLIW c
         end if
517-4:end loop

518-5: Program end
```

FIG. 6

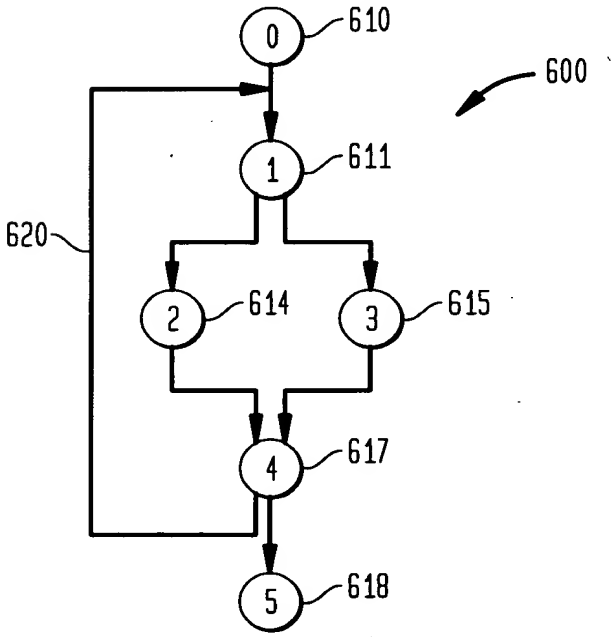


FIG. 7

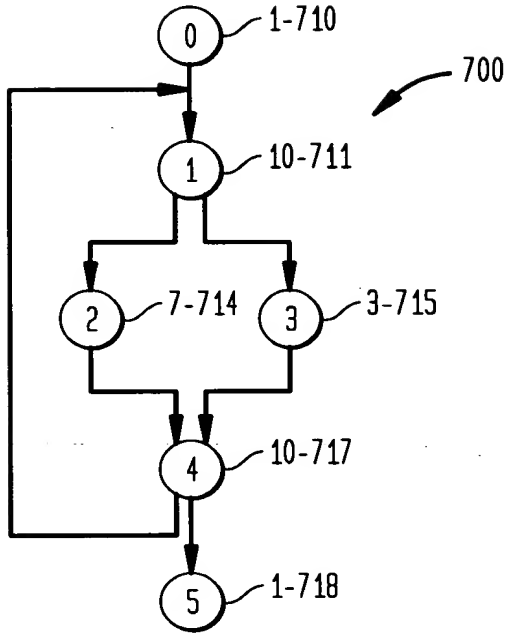


FIG. 8

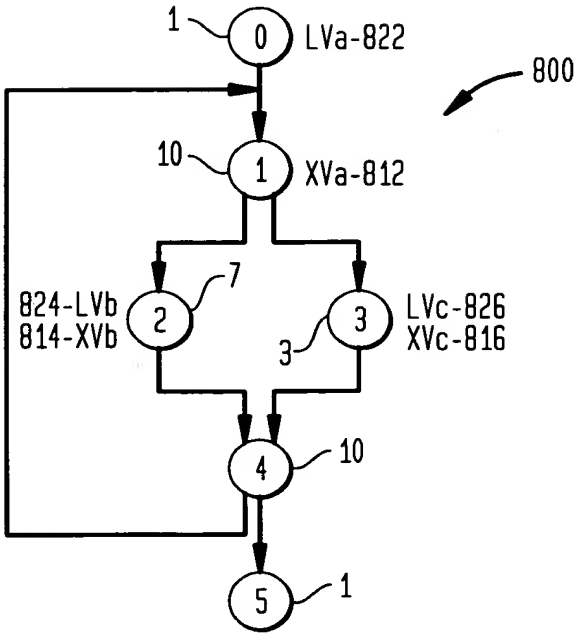


FIG. 9

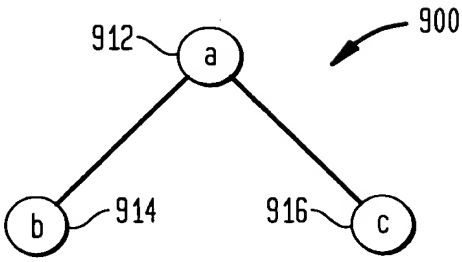
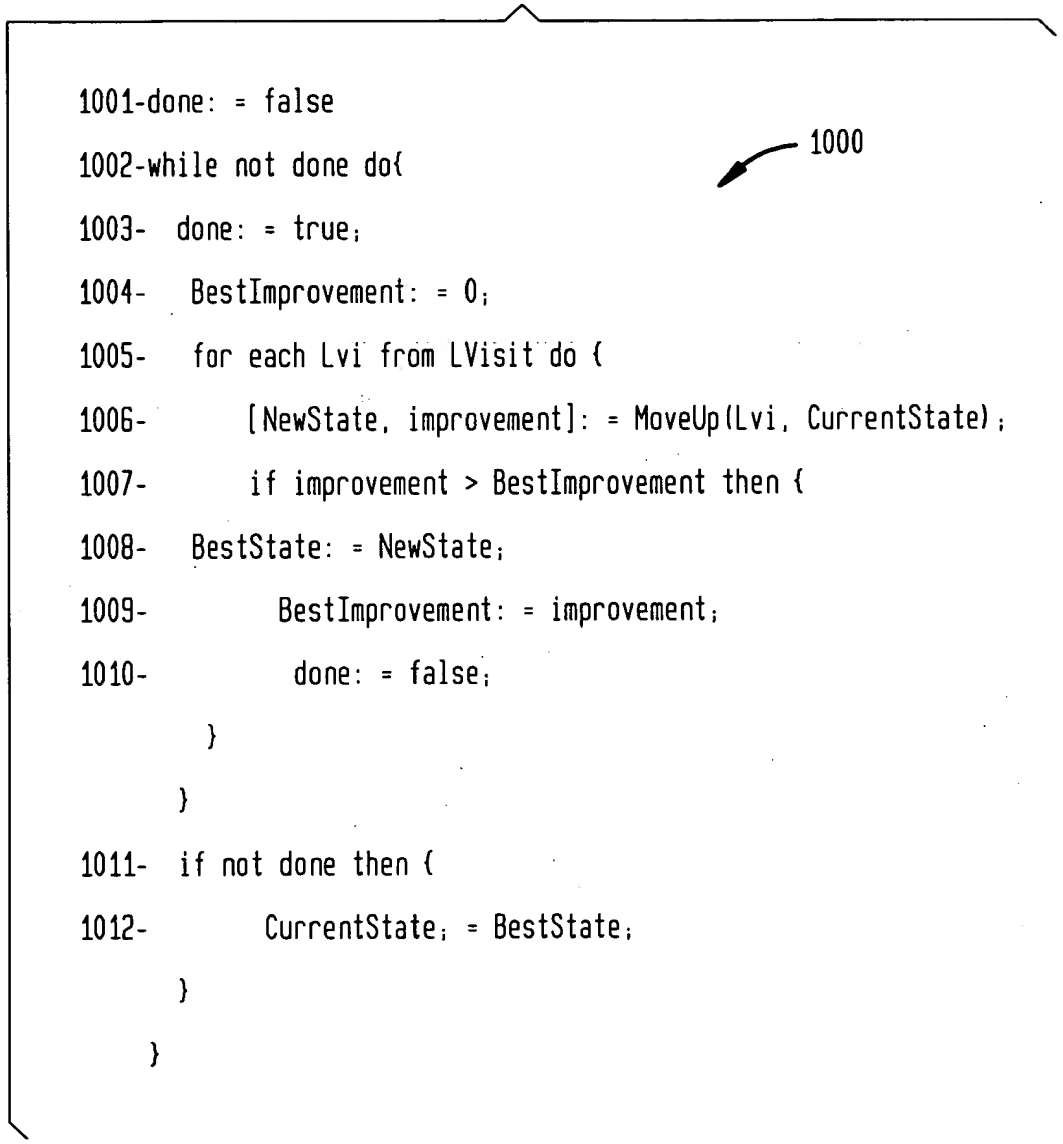


FIG. 10


1001-done: = false
1002-while not done do{
1000
1003- done: = true;
1004- BestImprovement: = 0;
1005- for each Lvi from LVisit do {
1006- [NewState, improvement]: = MoveUp(Lvi, CurrentState);
1007- if improvement > BestImprovement then {
1008- BestState: = NewState;
1009- BestImprovement: = improvement;
1010- done: = false;
}
}
1011- if not done then {
1012- CurrentState; = BestState;
}
}